Database Project Part 3:

Buxiao Chu, bc3730

Xingyu Xian xx2360

Yibo Zhang, yz10589

Languages: Python, SQL, HTML

Framework: Flask, Jinja2

Features chosen: 5, 6, 7, 8, 12

Schema

Our schemas are based on Project Schema-v2

Due to the implement of Feature 12 and other features, we change the schemas as follows:

Item

Item(**ItemID**, quantityNum, iDescription, photo, color, isNew, hasPieces, material, mainCategory, subCategory)

Item(mainCategory, subCategory) REFERENCES Category(mainCategory, subCategory)

Added `quantityNum` to it but not as a primary key.

When the same item added to the database, we just increase its quantity.

When donating, the staff will check if two items are the same so that they should have the same itemID.

Here is an assumption: We just suppose that the same kind of pieces are stored in the same location. If they have different locations, we will face a complex dispatch problem when shopping and delivering.

No AUTO_INCREMENT on `ItemID`

DonatedBy

DonatedBy(**ItemID**, quantityNum, **userName**, **donateDate**)

DonatedBy(ItemID) REFERENCES Item(ItemID)

DonatedBy(userName) REFERENCES Person(userName)

Added `quantityNum`to it but not as a primary key. Set donateDate as a primary key.

Because of the possible duplicated items, the same person might donate the same item in different date.

We just add this record into Donatedby. It has no effect on shopping.

Changed primary key to (**ItemID**, quantityNum, **userName**, **donateDate**)

ItemIn

ItemIn(**ItemID**, **quantityNum**, **orderID**, found, status, holdingRoomNum, holdingShelfNum)

ItemIn(ItemID) REFERENCES Item(ItemID)

ItemIn(orderID) REFERENCES Ordered(orderID)

ItemIn(holdingRoomNum, holdingShelfNum) REFERENCES Location(roomNum, shelfNum)

Added `quantityNum` to it as a primary key to display different locations.

Added `holdingRoomNum` for holding location for item

Added `holdingShelfNum` for holding location for item

Added `status` to view item status (for holding location feature)

Delivered

status ENUM('Pending', 'In Transit', 'Delivered', 'Cancelled') NOT NULL,
date DATE NOT NULL

Added `status`

Pages:

Index, login, register - the same as the demo

person - the home page, several functional links

find_item - show the information of the item's pieces

find_order - show the information of the order's items

donor_got - input of the information of a item from the user

item_added - input of the information of relevant pieces (if exists) from the user

start_order - Only staff person can create order for client person -- check username and role requirement.

add_to_order - add items into the new order from the remaining items list and update the quantity number.

prepare_order - input of the information of holding location of an order

user_orders - show all the orders the current user

Queries

Feature 1:

```sql
# login
SELECT * FROM Person WHERE userName = %s and password = %s

# register
# check if the user exists
SELECT * FROM Person WHERE userName = %s
# insert the user into database
INSERT INTO Person VALUES(%s, %s, %s, %s, %s)
insert into PersonPhone values(%s, %s)
insert into act values(%s, %s)
```

Feature 2:

```sql
# find item
select * from piece where itemID = %s
```

Feature 3:

```sql
# find order
# select items from the order
select * from ItemIn where orderID = %s
# count the item in the order
select count(*) as count from ItemIn where orderID = %s
# select the pieces of the items
select * from piece where itemID = %s
```

Feature 4:

```sql
# accept donation
# check if the user is a staff
select * from act where username = %s
# check if the user entered is a donor
select * from act where username = %s

# add item
# check if the item exists
select * from item where itemID = %s
# if it exists, update the quantity of this item
update item set quantityNum = quantityNum + %s where itemID = %s
insert into donatedby values(%s, %s, %s, %s)
# if it doesn`t exist, insert it into database
insert into item values(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
insert into donatedby values(%s, %s, %s, %s)

# add pieces
# if the item has pieces, insert them into database
insert into piece values(%s, %s, %s, %s, %s, %s, %s, %s, %s)
# show the pieces that have been added
select * from piece where itemID = %s
```

Feature 5:

```
# Check username and role requirement
SELECT 1 FROM Act WHERE userName = %s AND roleID = 'staff'
SELECT 1 FROM Act WHERE userName = %s AND roleID = 'client'

# Create new order
INSERT INTO Ordered (orderDate, orderNotes, supervisor, client) VALUES
(CURDATE(), %s, %s, %s
```

Feature 6:

```
# Create a drop down menu for staff to choose items from
mainCategory/subCategory
SELECT mainCategory, subCategory FROM Category ORDER BY mainCategory,
subCategory"

# Show item's information after select the category
SELECT i.ItemID, i.iDescription, i.quantityNum
FROM Item i
WHERE i.mainCategory = %s AND i.subCategory = %s AND i.quantityNum > 0

# Display the chosen item's quantity
SELECT quantityNum FROM Item WHERE ItemID = %s"

# If there exists an itemIn list that has the same itemID, update that
itemIn
SELECT quantityNum FROM ItemIn WHERE ItemID = %s AND orderID = %s",
UPDATE ItemIn SET quantityNum = quantityNum + %s WHERE ItemID = %s AND
orderID = %s"

# If not, create a new itemIn
INSERT INTO ItemIn (ItemID, orderID, quantityNum, found, status) VALUES (%s,
%s, %s, FALSE, 'Holding')"

# Update the remaining item quantity after adding
UPDATE Item SET quantityNum = quantityNum - %s WHERE ItemID = %s",

# Check the remaining item quatity is greater than 0, otherwise is sold
out/no left/don't display
SELECT i.ItemID, i.iDescription, i.quantityNum
FROM Item i
WHERE i.mainCategory = %s AND i.subCategory = %s AND i.quantityNum > 0

```

Feature 7:

```sql
# Search by Client Username
SELECT o.orderID, o.orderDate, o.orderNotes
FROM Ordered o
JOIN Person p ON o.client = p.userName
WHERE p.userName = %s

# Search by orderid
SELECT * FROM Ordered WHERE orderID = %s

# Display Order Details
SELECT i.ItemID, i.quantityNum, i.status, l.roomNum, l.shelfNum, l.shelfDescription
FROM ItemIn i
LEFT JOIN Location l ON i.holdingRoomNum = l.roomNum AND i.holdingShelfNum = l.shelfNum
WHERE i.orderID = %s

# Check current location and status of the item
SELECT holdingRoomNum, status
FROM ItemIn
WHERE orderID = %s AND ItemID = %s AND quantityNum = %s

# Update item location and status
UPDATE ItemIn
SET holdingRoomNum = %s, holdingShelfNum = %s, status = %s
WHERE orderID = %s AND ItemID = %s AND quantityNum = %s

# Fetch updated data
SELECT i.ItemID, i.quantityNum, i.status, l.roomNum, l.shelfNum, l.shelfDescription
FROM ItemIn i
LEFT JOIN Location l ON i.holdingRoomNum = l.roomNum AND i.holdingShelfNum = l.shelfNum
WHERE i.orderID = %s
ORDER BY i.ItemID, i.quantityNum

```

Feature 8:

```sql
# Fetch all orders associated with the current user
SELECT o.orderID, o.orderDate, o.orderNotes, o.supervisor, o.client, i.ItemID, i.quantityNum, i.status
FROM Ordered o
LEFT JOIN ItemIn i ON o.orderID = i.orderID
WHERE o.client = %s OR EXISTS (
        SELECT 1 FROM Delivered d WHERE d.orderID = o.orderID AND d.userName = %s
        )
```

Difficulties

1. We suppose that the same items are stored in the same location. It's known that in the real Logistics and transportation system, here`s a dispatch problem. The system needs to choose a appropriate warehouse and assign a delivery for clients. But we cant handle this.
2. We haven't implement photo uploading. we don't know how to handle them with the database.
3. We only used one branch to commit, so the changes for each person is hard to track since if one keeps committing, one will have to fetch from origin first but there are changes in the file. One of us accidentally pulled from origin and lost the recent worl, so we had to write the code again. A big lesson learned is to create different branches and keep committing for record. Then we can merge to main branch at the end.

Division of Work

Buxiao Chu

frame construction

provide data

feature 1,2,3,4 (12) coding, testing

Xingyu Xian

schemas revision

edit data

feature 7,8 (12) coding, testing

Yibo Zhang

provide data

feature 5,6 (12) coding, testing